Collaborative Filtering for Multi-class Data Using Belief Nets Algorithms

Xiaoyuan Su
Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
xsu@fau.edu

Taghi M. Khoshgoftaar
Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
taghi@cse.fau.edu

Abstract

As one of the most successful recommender systems, collaborative filtering (CF) algorithms can deal with high sparsity and high requirement of scalability amongst other challenges. Bayesian belief nets (BNs), one of the most frequently used classifiers, can be used for CF tasks. Previous works of applying BNs to CF tasks were mainly focused on binary-class data, and used simple or basic Bayesian classifiers [1][2]. In this work, we apply advanced BNs models to CF tasks instead of simple ones, and work on real-world multi-class CF data instead of synthetic binary-class data. Empirical results show that with their ability to deal with incomplete data, extended logistic regression on naïve Bayes and tree augmented naïve Bayes (NB-ELR and TAN-ELR) models [3] consistently perform better than the state-of-the-art Pearson correlation-based CF algorithm. In addition, the ELR-optimized BNs CF models are robust in terms of the ability to make predictions, while the robustness of the Pearson correlation-based CF algorithm degrades as the sparseness of the data increases.

1. Introduction

Collaborative filtering (*CF*) techniques use a database of user preferences for items to predict additional topics or products a new user might like. In a typical *CF* scenario, there is a list of m users $\{U_1, U_2, ..., U_m\}$ and a list of n items $\{I_1, I_2, ..., I_n\}$. Each user U_i has a list of items Iu_i on which the user has expressed his/her preferences or ratings, or simply the binary purchased/unpurchased or like/dislike. *CF* algorithms represent the entire $m \times n$ data as a user-item ratings matrix. Each value of $r_{i,j}$ in the matrix represents the rating score of the i-th user on the j-th item. There is an active user for whom collaborative filtering algorithms provide predictions or recommendations. Prediction represents the predicted preference on an item for the active user. Recommendation is a list of items that the active user will most likely prefer.

Collaborative filtering is one of the most successful recommendation techniques to date. However, collaborative filtering tasks face many challenges, especially for large online systems. The data for *CF* tasks are extremely sparse (with a very high rate of missing values) as each of the users can only purchase or rate a small percentage of items. *CF* algorithms are also required to have the ability to scale with an increasing number of users and items, to make satisfactory recommendations in a short time period, and to deal with other problems like synonymy, which refers to the tendency that the same or similar items to have different names.

Collaborative filtering techniques can be classified into three categories. *Memory-based (*or *correlation-based) CF* techniques use the user database to calculate the similarity or weight, $w_{i,j}$, between users or items and make predictions or recommendations according to those calculated similarity values. *Model-based CF* techniques use the user database to estimate or learn a model to make predictions [2]. The model can be a data mining and machine learning algorithm, such as Bayesian belief nets, association rules, or neural networks. *Hybrid CF* techniques combine two or more recommendation techniques to make predictions or recommendations, usually between *CF* and content-based filtering methods, which make recommendations by analyzing the content of textual information and finding regularities in the content.

Bayesian belief nets (*BNs*) classifier is one of the most frequently used classifiers. *BNs* models

can be applied to *CF* tasks as model-based *CF* algorithms. Previous work of applying *BNs* to *CF* tasks mainly used simple Bayesian models such as naïve Bayes model [1][4] and baseline Bayesian model [5] on binary datasets. Real-world *CF* data have more multi-class datasets than binary ones. For example, the three most used real-world *CF* databases: *MovieLens*, *EachMovie* and *Jester* are all multi-class data. We thus apply *BNs CF* models on multi-class data in this work, with a simple *BNs CF* model as well as two advanced ones. For the advanced *BNs CF* models, we apply the extended logistic regression (*ELR*) for naïve Bayes (*NB*) and Tree Augmented Naive Bayes (*TAN*) and call the resulting *CF* algorithms the *NB-ELR CF* and *TAN-ELR CF* models. *ELR* is a discriminative parameter-learning algorithm that maximizes log conditional likelihood (*LCL*) for the fixed Bayesian belief net, *NB* or *TAN*, and has high classification accuracy for both complete data and incomplete data [3].

As drawing comparative and convincing conclusions from synthetic datasets is risky because the data may fit one of the algorithms better than others, we work on subsets of the real-world data from *MovieLens* [6]. *MovieLens* is a web-based movies recommender system with 43,000 users and their ratings for over 3,900 movies.

Instead of *classification accuracy* or *classification error*, the most widely used evaluation metric for *CF* is the Mean Absolute Error *(MA*E), which is the average of the absolute difference between predictions and user-specified values [7].

$$MAE = \frac{\sum_{j=1}^{N} |p_{ij} - r_{ij}|}{N}$$

where N is the number of items user i has rated, p_{ij} is the predicted rating for user i on item j, r_{ij} is the actual rating. *MAE* sums the absolute errors of the N corresponding rating-prediction pairs $<p_{ij}, r_{ij}>$ and then computes the average. The lower the *MAE*, the better the prediction.

Other evaluation metrics include variations of *MAE* such as *NMAE* and *RMSE*, as well as *precision, recall*, *F1 metric, Reversal rat*e, *ROC sensitivity, PRC sensitivity*, *Half-life utility metric* and *NDPM* etc [8]. We use *MAE* as the performance criterion in this paper.

In section 2, we present the commonly-used correlation-based collaborative filtering algorithms, and propose and discuss the *robustness* of *CF* algorithms. In section 3, we present collaborative filtering using *BNs,* including a simple *BNs CF* algorithm (*naïve Bayes*) and two advanced *BNs CF* algorithms (*NB-ELR CF* and *TAN-ELR CF* models), both on multi-class data. Experimental design and results are in Section 4 and Section 5 respectively, and conclusions in Section 6.

2. Correlation-based Collaborative Filtering Algorithms

Correlation-based *CF* algorithms use the entire or a sample of the user-item database to generate a prediction. They first calculate the similarity or weight, $w_{i,j}$, which reflects distance, correlation, or weight, between two users or two items, i and j. Next they produce a prediction for the active user by taking the weighted average of all the ratings of that user on a certain item, or using a simple weighted average [9]. A commonly used similarity is *Pearson correlation*, which measures the extent to which two variables linearly relate with each other [10]. *Pearson correlation* between user i and j is given by

$$w_{i,j} = \frac{\sum_{u} (r_{i,u} - \bar{r}_i)(r_{j,u} - \bar{r}_j)}{\sqrt{\sum_{u} (r_{i,u} - \bar{r}_i)^2} \sqrt{\sum_{u} (r_{j,u} - \bar{r}_j)^2}}$$

where the summations over the subscript u are over the items which both the users i and j have rated. $\bar{r}_i$ is the average rating of the both-rated items of the i-th user, and the same for $\bar{r}_j$. Other similarity measures include variations of *Pearson correlation* and *vector cosine similarity* and its variations [9].

After calculating the similarities, we can make a prediction for a certain user, a, on a certain item, i, by taking a weighted average of all the ratings on that item according to the following formula [10].

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u} |w_{a,u}|}$$

where $\bar{r}_a$ and $\bar{r}_u$ are the average ratings for the user a and user u on all other rated items than i, $w_{a,u}$ is the similarity between the user a and user

u. The above process is called a user-based *CF* algorithm.

An item-based *CF* algorithm uses the similarity between co-rated items (i.e., cases where the user rated both the two items) and make predictions using a *simple weighted average* to predict the rating $P_{a,i}$ for user a on item I

$$P_{a,i} = \frac{\sum_u r_{a,u} w_{i,u}}{\sum_u |w_{i,u}|}$$

where the summations are over all items u similar to i by user a, $w_{i,u}$ is the similarity between items i and u. $r_{a,u}$ is the rating for user a on item u.

	matrix of CF ratings				*Correlation-based CF prediction*			
	I_1	I_2	I_3	I_4	I_1	I_2	I_3	I_4
U_1	4	**3**	5	5	2.53	**3.95**	2.93	4.75
U_2	4	2	1		2.17	4.50	3.67	
U_3	3		2	4	2.00		4.17	0.50
U_4	4	4			?=3	?=3		
U_5	2	1	3	5	2.90	1.80	1.81	1.75
					MAE=1.56			

(a) **(b)**

Table 1 (a) an example of matrix of ratings (b) the corresponding predictions from correlation-based CF algorithm

For example in Table 1 (a), using the user-based *CF* algorithm to predict the rating for U_1 on I_2, we have

$$P_{1,2} = \bar{r}_1 + \frac{\sum_u (r_{u,2} - \bar{r}_u) \cdot w_{1,u}}{\sum_u |w_{1,u}|}$$

$$= \bar{r}_1 + \frac{(r_{2,2} - \bar{r}_2) w_{1,2} + (r_{4,2} - \bar{r}_4) w_{1,4} + (r_{5,2} - \bar{r}_5) w_{1,5}}{|w_{1,2}| + |w_{1,4}| + |w_{1,5}|}$$

$$= 4.67 + \frac{(2 - 2.5)(-1) + (4 - 4)0 + (1 - 3.33)0.756}{1 + 0 + 0.756} = 3.95$$

We have predictions from the *Pearson correlation*-based *CF* algorithm for each of the observed values, and we then calculate the *MAE* (Table 1(b)).

From this example, we can find that the correlation-based *CF* algorithm can not make predictions for $r_{4,1}$ and $r_{4,2}$ because the denominator (the summation of the similarities) of the prediction function is 0. In this situation, we can use the *default voting* as the prediction, which can be either the average rating of that user on all of his/her rated items, or the *universal average rating* (3 in datasets of our work), and is actually not from the similarity calculation. Using the *universal average rating* instead of the average rating of the user gives better performance because a large percentage of failures in making predictions happen when the average rating of the user is 0, and using 3 instead of 0 gives a better *MAE*. The contribution from *default voting* to the *MAE* is taken into the performance evaluation of our *CF* algorithms, because we need to compare the performances over the same number of predictions.

We estimate the impact of the *default voting* to the *MAE* in terms of *robustness* of the algorithms, which can be defined as the number of predictions made using the algorithms (without using *default voting*) divided by the total number of predictions that should be made.

$$Robustness = \frac{\#(predictionsMade)}{\#(predictionsNeedToBeMade)}$$

For the correlation-based *CF* algorithm, the zero-value denominator of the prediction calculation degrades the *robustness* of the algorithm.

Although simple to implement, the *Pearson correlation*-based *CF* algorithm is one of the state-of-the-art *CF* techniques.

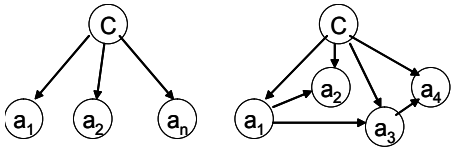
3. Collaborative Filtering Using Bayesian Belief Nets

Bayesian belief nets (*BNs*) are often used for classification tasks. Motivated by the simplicity and accuracy of the naïve Bayes (*NB*) classifier, *BNs* are increasingly used for pattern recognition, fault diagnosis and other classification tasks.

A Bayesian belief net (*BN*) is a directed, acyclic graph (*DAG*) with a probabilistic graph model $B = \langle N, A, \Theta \rangle$, where each network node $n \in N$ represents a random variable and each directed arc $a \in A$ between nodes represents a probabilistic association between variables, and Θ represents a conditional probability table (*CPtable*) quantifying how much a node depends on its parents.

A *NB* network has a simple structure with the class node as the parent of all the attribute nodes. No connections between attribute nodes are allowed in a *NB* structure (Figure 1(a)). A *TAN*

network includes a link from the class node down to each attribute and, if we ignore those class-to-attribute links, the remaining links connect attributes to each other and form a tree (Figure 1(b)) [11]. *TAN* has a more complex *BN* structure and generally has better classification performance than the simpler structured *NB*.



(a) **(b)**

Figure 1 (a) NB structure (b) TAN structure

BNs classifiers can make classifications for both multi-class data and binary-class data. As high sparsity of data is a key characteristic of *CF* tasks, the ability to handle the high-missing-rate incomplete data is required for *BNs CF* models.

3.1 Naïve Bayes CF Algorithm

The naïve Bayes model uses an *NB* classifier as its classification model for collaborative filtering tasks. Assuming the features are independent given the class, the probability of a certain class given all of the features $p(C_j|f_1,f_2,...f_n)$ can be found by computing $p(C_j)\prod_i^n p(f_i \mid C_j)$, where both $p(C_j)$ and $p(f_i|C_j)$ can be estimated from training data (C_j refers to class j, f_i refers to feature i), the class with the highest probability will be classified as the predicted class. For incomplete data, the probability calculation and classification production are computed over observed data (the subscript o in the following equation indicates observed values), which is an effective way to handle missing values when there are enough observed data to make reliable classifications.

$$class = \arg\max_{j \in classSet} p(class_j) \prod_o p(X_o = x_o \mid class_j)$$

The *Laplace Estimator* can be used to smooth the probability calculation and avoid a conditional probability of 0.

$$P(X_i = x_i \mid Y = y) = \frac{\#(X_i = x_i, Y = y) + 1}{\#(Y = y) + |X_i|}$$

where $|X_i|$ is the size of the set $\{X_i\}$. For an example of binary class, $P(X_i=0|Y=1)$=0/2 will be (0+1)/(2+2)=1/4, $P(X_i=1|Y=1)$=2/2 will be (2+1)/(2+2)=3/4 using the *Laplace Estimator*.

Previous work of applying *BNs* to *CF* tasks is mainly focused on binary-class data. For example, in [4], multi-class data are first converted to binary-class data, and then converted to a Boolean feature transformation of the ratings matrix, doubling the user numbers by transforming each user U_n to U_n*like* and U_n*dislike*. The rating of 1 for U_n is converted to 1/0 corresponding to U_n*like*/U_n*dislike*, and 0 and missing value converted to 0/1 and 0/0 respectively. These conversions facilitate the use of the *NB* algorithm for *CF* tasks, but bring the loss of multi-class information and increase the burden of scalability, especially for multi-class data, e.g., 5-class data needs 5 times more user numbers by simply using the Boolean feature transformation of the ratings matrix, which will not be realistic for real-world *CF* tasks. In [1], they applied the *NB CF* model only on binary data.

In our work, we apply the *NB CF* algorithm directly to real-world multi-class data for *CF* tasks and produce straightforward predictions for users.

For the same example in Table 1(a), to predict the rating for U_1 on I_2 using the *NB CF* algorithm and the *Laplace Estimator*, we have

$$\begin{aligned} class &= \arg\max_{c_j \in \{1,2,3,4,5\}} p(c_j \mid U_2 = 2, U_4 = 4, U_5 = 1) \\ &= \arg\max_{c_j \in \{1,2,3,4,5\}} p(c_j) p(U_2 = 2 \mid c_j) p(U_4 = 4 \mid c_j) p(U_5 = 1 \mid c_j) \\ &= \arg\max_{c_j \in \{1,2,3,4,5\}} \{0 \ , \ 0 \ , \ 0 \ , \ 0.0031 \ , \ 0.0019 \ \} \\ &= 4 \end{aligned}$$

in which $p(4)p(U_2=2|4)p(U_4=4|4)p(U_5=1|4)$ = (1/3)*(1/6)*(2/6)*(1/6) = 0.0031.

For the *NB CF* algorithm, there are cases where the maximum probability is 0 when calculating the prediction, and therefore one can not produce predictions using the algorithm. We also use the *default voting* as the prediction in this situation. The zero-value maximum probability of the prediction calculation for the *NB CF* algorithm degrades the robustness of the algorithm.

3.2 NB-ELR and TAN-ELR CF Models

As computing the optimal *CPtable* entries is generally intractable, *Greiner et al.* proposed a gradient-ascent algorithm, extended logistic

regression (*ELR*), which is a discriminative parameter-learning algorithm that maximizes *log conditional likelihood* [3].

$$\hat{LCL}^{(S)}(\Theta) = \frac{1}{|S|} \sum_{<e_i,c_i>\in S} \log(P_\Theta(c_i \mid e_i))$$

where S is the sample space $\{<c_i,e_i>\}$, and each class label c_i is associated with evidence e_i. *ELR* extends standard logistic regression (*LR*) [12] to accommodate incomplete training data, while most *LR* algorithms only require complete data.

ELR optimizes the *log conditional likelihood* $\hat{LCL}(\Theta)$ by changing the values of each *CPtable* entry $\Theta_{d|f}$. To incorporate the constraints $\Theta_{d|f} \geq 0$ and $\Sigma_d \Theta_{d|f}=1$, they used a different set of parameters: the logistic $\beta_{d|f}$, where

$$\Theta_{d|f} = \frac{e^{\beta_{d|f}}}{\sum_{d'} e^{\beta_{d'|f}}}$$

As the β_i values sweep over the reals, the appropriate constraints will be satisfied by the corresponding $\Theta_{d|f}$ values. *ELR* is basically

Initialize $\beta^{(0)}$
For k=1…m
$\beta^{(k+1)} = \beta^{(k)} + \alpha^{(k)} \times d^{(k)}$

where $\beta^{(0)}$ is the initial plug-in parameters, $\beta^{(k)}$ is the set of parameters at iteration k, $\alpha^{(k)}$ is the magnitude of the changes calculated with line search, $d^{(k)}$ is the direction of the modification (conjugate gradient), and m is the stopping criteria called cross tuning [3]. Given a set of labeled queries, *ELR* descends to the direction of the total derivative with respect to these queries, which is the sum of the individual derivatives.

ELR uses observed frequency estimates (*OFE*) [17] to initialize the parameters, $\beta^{(0)}$. These easy-to-compute generative starting values are often used to initialize parameters for discriminative tasks [13]. It uses the *conjugate gradient* method to descend along conjugate directions, rather than simply the local gradient, and requires far fewer steps to reach the local optimum [14]. A standard *Brent*'s *iterative line search procedure* [14] is used to decide how far *ELR* will ascend in the $d^{(k)}$ direction. *Cross tuning* is used in *ELR* to estimate the optimal number of iterations [3].

In [3], working on 20 incomplete datasets from the *UCI* machine learning repository with different missing rates, mostly between 0.06% and 30.17% and one with 64.94%, and using the *classification accuracy* criterion, *NB-ELR* performs significantly better than *NB-APN* and *NB-EM*. *NB-APN* is the *NB* optimized by a standard missing-data learning algorithm, Adaptive Probabilistic Networks (*APN*) [15], and *NB-EM* is by Expectation Maximization (*EM*) [16], both of which ascend to parameter values whose likelihood is locally optimal.

NB-ELR performs similarly to or slightly worse than *TAN-ELR*, in which the *BNs* classifier *TAN* is optimized by *ELR*. *NB,* however, has a simpler structure and *NB-ELR* can be learned in a much shorter time period than *TAN-ELR*. We implement both *NB-ELR* and *TAN-ELR* for the *CF* tasks in this work and compare them with other *CF* algorithms.

3.3 Other Bayesian Belief Nets CF Models

There are some other *BNs* models for *CF* tasks in *CF* research literature, such as the *BNs with decision trees model***,** which has a decision tree at each node of the *BN*, with a node corresponds to each item in the domain and the states of each node correspond to the possible ratings for each item [2]; and the *Baseline Bayesian model*, which uses a Bayesian belief net with no arcs (baseline model) for collaborative filtering and recommends items on their overall popularity [5].

4. Experimental Design

We implemented a commonly-used correlation-based *CF* algorithm (user-based *CF* algorithm using the *Pearson Correlation* as the similarity value), a simple *BNs CF* algorithm (*NB CF* algorithm) and two advanced *BNs CF* algorithms (*NB-ELR CF* model and *TAN-ELR CF* model) to compare performances of the *CF* algorithms.

We evaluated 17 subsets of the real-world *MovieLens* data, with different missing rates from 53.8% to 97.2%. Each of the subset datasets has observed ratings for 943 users on 20 movies, with the rating values from 1 to 5. The original *MovieLens* dataset used has 100,000 ratings for 1682 movies by 943 users.

We also worked on three other subsets with the same number of users and items and with missing rates of 98.1%, 98.9% and 99.5% respectively. However, because the *robustnesses* of correlation-based *CF* algorithm and *NB CF* algorithm are very low, (e.g., correlation-based

CF algorithm can only make 44, 8 and 0 predictions for the three datasets for 358, 207 and 94 need-to-be-predicted values respectively), the overall *MAE* will come more from a *default voting* than from the algorithms. We therefore do not count the *MAE*s from these three datasets in our overall performance comparison.

For the correlation-based *CF* algorithm and naïve Bayes *CF* algorithm, we assume each of the observed values were missing and predict a rating value for it. Then we calculate the difference between the actual value and predicted value, and continue to work on the next observed value and finally sum up the absolute errors and obtain the Mean Average Error (*MAE*) to evaluate the performance of the algorithms. For each new *CF* task, it just needs to calculate its related similarities or conditional probabilities to produce predictions from the algorithms. New evidences of ratings are incrementally incorporated with the existing ones.

For the *NB-ELR* and *TAN-ELR CF* models, we use each column of the rating matrix in turn as the class column and the remaining columns as the attributes, remove the users with missing values on the class column and use 5-fold cross-validation to train and test the *NB-ELR* and *TAN-ELR* models. We use *OFE* [17] as the type of the *CPtable* initialization, use 5-fold cross-tuning and 20 as the maximum iteration number for the *NB-ELR* and *TAN-ELR* training, which are the options required by the *ELR* algorithm [3]. The models produce predictions and calculate the *MAE* for each dataset. Then we use each of the other columns as the class column in turn and repeat the above steps and get the average *MAE* of all testing results. For each new *CF* task, i.e., to predict a rating of a certain user on a certain item, the prediction is available from the existing trained model. When necessary (e.g., the system has gotten a large amount of new evidences), new evidences can be re-trained together with existing ones to give an updated model.

5. Results

Using real-world multi-class *CF* datasets, the empirical results show that the correlation-based *CF* algorithm, *NB-ELR CF* algorithm, and *TAN-ELR CF* algorithm perform significantly better than the *naïve Bayes CF* algorithm. Overall, in terms of *MAE*, *NB-ELR CF* and *TAN-ELR CF* algorithms have consistently better performances than the correlation-based *CF* algorithm. Statistically, using paired *T-Test*, *NB-ELR CF* is better than the correlation-based *CF* algorithm with p=0.0002, *TAN-ELR CF* is better than correlation-based *CF* algorithm with p=0.00004, and it is also better than *NB-ELR CF* with p=0.0077 (see Table 2 and Figure 2).

For the response time, the *NB CF* algorithm produces all the predictions in the shortest time of the four algorithms: within 2 seconds, averaged over the 17 datasets. It takes 5 seconds on average for the correlation-based *CF* algorithm to produce all the predictions, 3.7 minutes for the *NB-ELR CF* algorithm and 12 minutes for the *TAN-ELR CF* algorithm to train and test each dataset using computers with AMD Athlon XP 1.1GHz processors and 1GB memory. The training and prediction time of *NB-ELR CF* is acceptable when it's not very time-critical. Although the *TAN-ELR CF* algorithm produces the best predictions, it's not desirable for time-critical situations. One solution is to run the time-consuming training stage offline, and the online prediction-producing stage will take a much shorter time.

The fast degradation of the performances of all four algorithms when the missing rate is higher than 90% is due to the lack of enough observed values to make predictions.

Sparsity (%)	correlation-based CF	NB CF	nb-elr CF	TAN-ELR
53.8	0.796	1.071	**0.769**	0.780
56.8	0.840	1.170	**0.822**	0.827
59.9	0.823	1.122	0.799	**0.797**
62.2	0.795	1.095	0.782	**0.770**
66.8	0.810	1.109	0.787	**0.787**
69.0	0.798	1.059	0.773	**0.765**
72.5	0.841	1.191	0.822	**0.798**
75.1	0.832	1.113	**0.804**	0.821
79.0	0.872	1.237	0.824	**0.814**
81.8	**0.826**	1.197	0.860	0.853
83.9	0.843	1.087	0.845	**0.818**
86.1	0.903	1.169	0.887	**0.839**
89.0	0.879	1.121	0.857	**0.831**
91.1	0.921	1.100	0.876	**0.831**
93.7	0.917	1.031	0.841	**0.820**
95.4	1.027	1.097	**0.960**	0.969
97.2	1.087	1.202	1.040	**1.022**
Average	0.871	1.128	0.844	**0.832**

Table 2, MAE of the four CF algorithms

In terms of robustness, the *NB-ELR* and *TAN-ELR CF* algorithms consistently produce predictions regardless of the data sparseness and obtain a perfect robustness of 1 for the datasets in this case study. The correlation-based *CF*

algorithm degrades faster than the *NB CF* algorithm when the missing rate of the dataset increases (Figure 3) and is more frequently unable to produce the predictions.

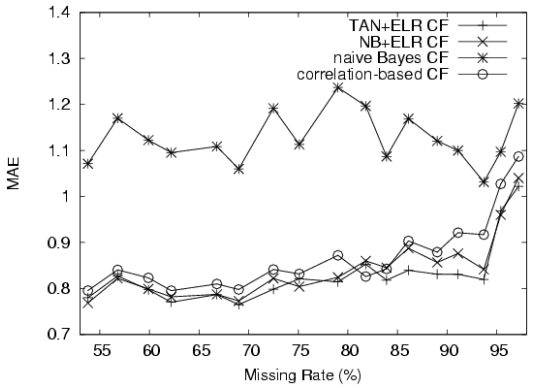


Figure 2, the performances of the four CF algorithms against missing rates of the data

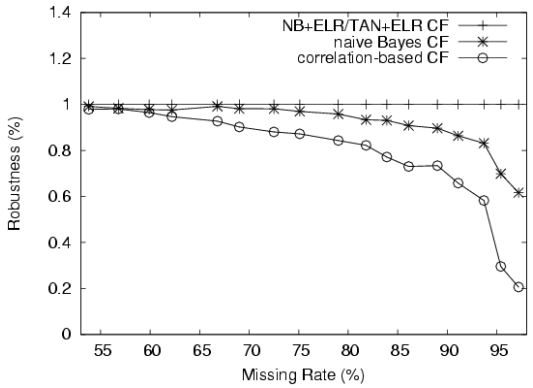


Figure 3, robustness of the four CF algorithms

6. Conclusions

Collaborative filtering (*CF*) is one of the most important recommendation systems. Algorithms with a strong ability to deal with sparsity, scalability and other challenges will be suitable for *CF* tasks. In this work, we apply Bayesian belief nets (*BNs*) algorithms on real-world multi-class *CF* datasets, and specifically, we apply advanced *BNs* models instead of simple ones in the *CF* realm. *NB-ELR* and *TAN-ELR,* the *NB* and *TAN BNs* optimized by the extended logistic regression (*ELR*) algorithm, have superior performance when dealing with incomplete data for *CF* tasks. Empirical results show that *NB-ELR* and *TAN-ELR* perform consistently better than the state-of-the-art *Pearson correlation*-based *CF* algorithm across the whole missing rate spectrum from 53.8% to 97.2%. A simple *BNs CF* algorithm, *NB CF* algorithm, performs worse than the correlation-based *CF* algorithm on multi-class data. In terms of *robustness* of the four algorithms, *NB-ELR* and *TAN-ELR CF* algorithms are very robust in producing predictions for *CF* tasks, while the *NB CF* algorithm and correlation-based *CF* algorithm become more frequently unable to make predictions with the increase of the missing rate of the data. The *robustness* of the correlation-based *CF* algorithm degrades faster than *NB CF* algorithm. Of the four algorithms, *TAN-ELR* performs the best in terms of *MAE*, but requires the longest time to train the model and make predictions. The second-best *NB-ELR* (with *MAE* 0.012 shy of *TAN-ELR*) will be more practical because it gives predictions more quickly.

Acknowledgement: We are grateful to the current and former members of the Empirical Software Engineering and Data Mining and Machine Learning Laboratories at Florida Atlantic University for their reviews and comments. We thank Dr. Russell Greiner for his valuable suggestions.

References

[1] K. Miyahara and M.J. Pazzani, Improvement of Collaborative Filtering with the Simple Bayesian Classifier, *Information Processing Society of Japan*, 43(11), 2002.

[2] J. Breese, D. Heckerman, C. Kadie, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, *Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence,* Madison, WI, Morgan Kaufmann, July 1998.

[3] R. Greiner, X. Su, B. Shen, W. Zhou, Structural Extension to Logistic Regression: Discriminative Parameter Learning of Belief Net Classifiers, *Machine Learning*, 59(3), pp. 297 – 322, 2005.

[4] K. Miyahara and M.J. Pazzani, Collaborative Filtering with the Simple Bayesian Classifier, *6th Pacific Rim International Conference on Artificial Intelligence*, Melbourne, Australia, pp.679-689, 2000.

[5] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, C. Kadie. Dependency, Networks for Density Estimation, Collaborative Filtering, and Data Visualization, *Journal of Machine Learning Research,* 1:49-75, 2000.

[6] http://movielens.umn.edu, GroupLens Research group, Department of Computer Science and Engineering, University of Minnesota.
[7] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: A Constant Time Collaborative Filtering Algorithm, *Information Retrieval,* 4(2), pp. 133-151, July 2001.
[8] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, Evaluating Collaborative Filtering Recommender Systems, *ACM Transactions on Information Systems*. Vol. 22. No.1, Jan pp5-53, 2004.
[9] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Riedl, Item-based Collaborative Filtering Recommendation Algorithms, *10th International World Wide Web Conference*, ACM Press, pp. 285-295, 2001.
[10] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, Grouplens: An open architecture for collaborative filtering of netnews, *Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work*, pp. 175-186, New York, 1994.
[11] N. Friedman, D. Geiger and M. Goldszmidt, Baysian Network Classifiers, *Machine Learning*, 29, pp. 131-163, 1997.
[12] P. McCullagh, and J. Nelder, *Generalized linear models*. Chapman and Hall, 1989.
[13] B. Ripley, *Pattern Recognition and Neural Networks,* Cambridge University Press, 1996.
[14] M. Hagan, H. Demuth, and M. Beale, *Neural Network Design*, PWS Publishing, 1996.
[15] J. Binder, D. Koller, S. Russell, and K. Kanazawa, Adaptive probabilistic networks with hidden variables. *Machine Learning, 29*, pp. 213–244, 1997.
[16] A, Dempster, N. Laird, and D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. Royal Statistics Society, Series B, 39*, 1977.
[17] G. Cooper, and E. Herskovits, A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*, pp. 309–347, 1992.